Integration of a Motion Capture System into a Spacecraft Simulator for Real-Time Attitude Control*

Benjamin L. Reifler
University at Buffalo, Buffalo, New York

1st Lt Dylan R. Penn
Air Force Research Laboratory, Kirtland Air Force Base, New Mexico

16 August 2016

Abstract

The Attitude Control System Proving Ground (ACSPG) is a three-degree-of-freedom spacecraft simulator mounted on a low-friction spherical air bearing. It is designed to test the behavior of integrated hardware and software for attitude determination and control systems (ADCS). These systems require external references, such as the positions of stars and the direction of the Earth's magnetic field, to maintain accurate attitude estimation, but on Earth, such references are difficult to emulate in hardware. Therefore, the ACSPG testbed has been integrated with a PhaseSpace Impulse X2 motion capture system. This system calculates the testbed's inertial attitude, which can be used to simulate various sensors and to validate attitude estimation performance. This paper describes the suite of software programs that has been developed to transmit, receive, log and analyze attitude and performance data. This software architecture makes it possible for the ACSPG's ADCS to wirelessly receive inertial attitude data in near real time and provides diagnostic tools to aid in debugging the system and characterizing its performance.

Contents

*This work was conducted while the first author was an intern at the Air Force Research Laboratory (AFRL) Space Vehicles Directorate.

DISTRIBUTION A. Approved for public release: distribution unlimited.

1 Introduction

The attitude determination and control subsystem (ADCS) is a key part of almost any spacecraft, responsible for using a combination of sensors, actuators and algorithms to direct the rotational motion of the vehicle. Common uses of ADCS include pointing solar panels at the Sun, radio antennas toward ground stations and sensors toward targets of scientific interest, among others[1]. Such systems are difficult to test on Earth because space is comparatively a zero-torque environment (neglecting solar radiation pressure and air drag), but air-bearing testbeds like the Attitude Control System Proving Ground (ACSPG) offer low-disturbance torque platforms for ADCS testing[2].

The ACSPG platform features an onboard inertial measurement unit (IMU), which can provide real-time angular rate and acceleration information to the flight computer. Rates and accelerations are not sufficient, however, to maintain attitude knowledge over time, as sensor errors build up and cause the attitude estimate to diverge from reality. In order to counter this effect, real spacecraft use a variety of sensors including magnetometers, star trackers and sun sensors to measure their attitude relative to known references (the Earth's magnetic field, the stars and the direction of the Sun, respectively). It is expensive and difficult to emulate magnetometers and star trackers on Earth[3], so instead the ACSPG must generate measurements via another source. To that end, a PhaseSpace Impulse X2 motion capture system has been integrated with the ACSPG and a wireless link has been established between the motion capture system and its onboard SpeedGoat Simulink Real-Time computer[4,5].

2 System Overview

The Impulse X2 motion capture system consists of a series of cameras on wall-mounted rails, which observe LEDs that blink at different frequencies mounted around the top surface of the moving ACSPG platform (Figure 1). These cameras feed data to a specialized server, the Owl server, where PhaseSpace software processes them into quaternions (four-dimensional numbers consisting of one real and three imaginary parts) representing the platform's orientation relative to an inertial frame of reference established by the PhaseSpace system.

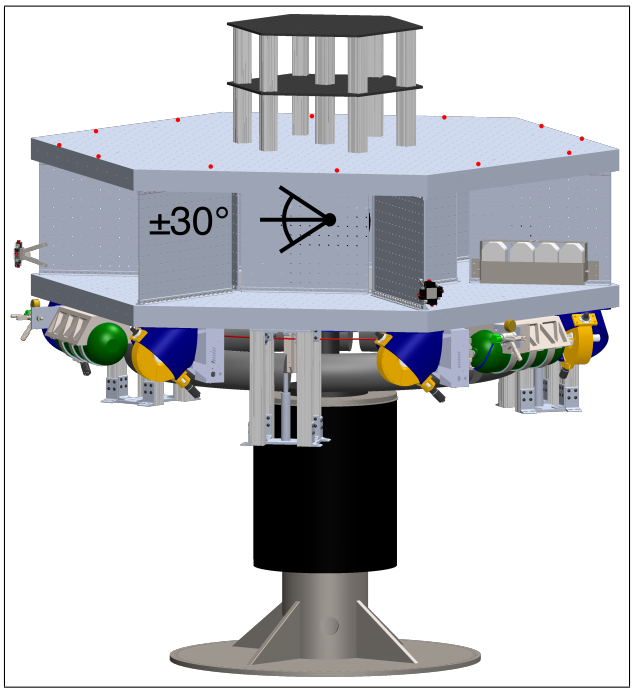


Figure 1: CAD drawing of the ACSPG, showing its ±30° range of roll/pitch. The 12 LEDs are marked in red.

The quaternion data (q) are routed wirelessly from the Owl server to the SpeedGoat computer on the ACSPG, while maintaining a low rate of packet loss and providing new packets in as close to real time as possible. In addition to providing real-time data to the ACSPG, it is also necessary that the system include diagnostic tools to help in characterizing performance and debugging potential problems or inconsistencies during future use.

A suite of software programs was developed to transmit, receive, log and analyze attitude and performance data. Figure 2 shows the flow of information through the final system, and Table 1 summarizes the software and its functions.

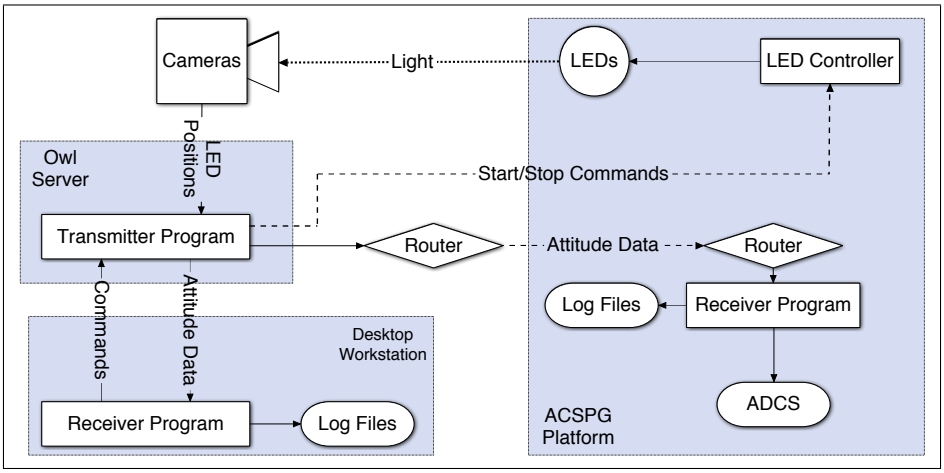


Figure 2: Information flow through the final system. Solid lines are physical connections, dashed lines are wireless.

Table 1: Software components.

Program	Function
Transmitter Program	Initializes PhaseSpace system, samples attitude data and transmits it to receivers
Simulink Receiver Program	Receives and logs data, distributes attitude data to ADCS components
Workstation Receiver Program	Receives and logs data, produces real-time plots
Plotting Script	Processes logged data and produces statistical plots

3 Software Design Process

3.1 Protocol Selection

The first decision to be made in designing the communication link between the Owl Server and the ACSPG's onboard computer was the selection of a communication protocol. The two protocols that were considered were the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP), which are compared in Table 2. The former provides reliable, in-order delivery of data packets, which is why it is used for internet communications that are expected to deliver complete, uncorrupted files, such as web browsing and email[6]. The latter, in contrast, features no additional checks to ensure that packets actually reach their destinations or that they arrive in sequence. Because UDP ignores corrupted and missing packets, has smaller headers and does not wait for an ACK (acknowledgment) message before sending more data, it is much faster than TCP and is generally preferred for real-time applications such as online gaming, video streaming and internet radio[6].

Table 2: Comparison of relevant features of TCP and UDP protocols.

Feature	TCP	UDP
Loss Recovery	Yes	No
Consistent Ordering	Yes	No
Requires Established Connection	Yes	No
Header Size (bytes)	20	4
Waits for ACK	Yes	No
Allows Broadcast/Multicast	No	Yes

The ACSPG needs to be able to receive attitude information that is as up-to-date as possible, which makes TCP a less attractive option due to larger packets and the potential for high time delays every time a packet fails to arrive intact[6]. Additionally, UDP could be used to create a system that incorporates TCP-like features, for example reducing loss by sending ACK messages (via UDP) and mitigating out-of-sequence data by rearranging packets after reading them from the network buffer. Based on these factors, UDP was chosen as the communication protocol. Since completing software development, test data (Figure 3) have shown that while UDP suffers some loss, the rate of lost packets is less than four percent at the lowest transmission rate allowed by the motion capture system (120 Hz) and less than seven percent at the maximum rate (480 Hz). These losses are acceptable for this application because the data are being sampled at a faster rate than is necessary for attitude estimation. Additionally, packets rarely arrive out of order, so any potential problem can be avoided simply by discarding those that have out-of-sequence timestamps.

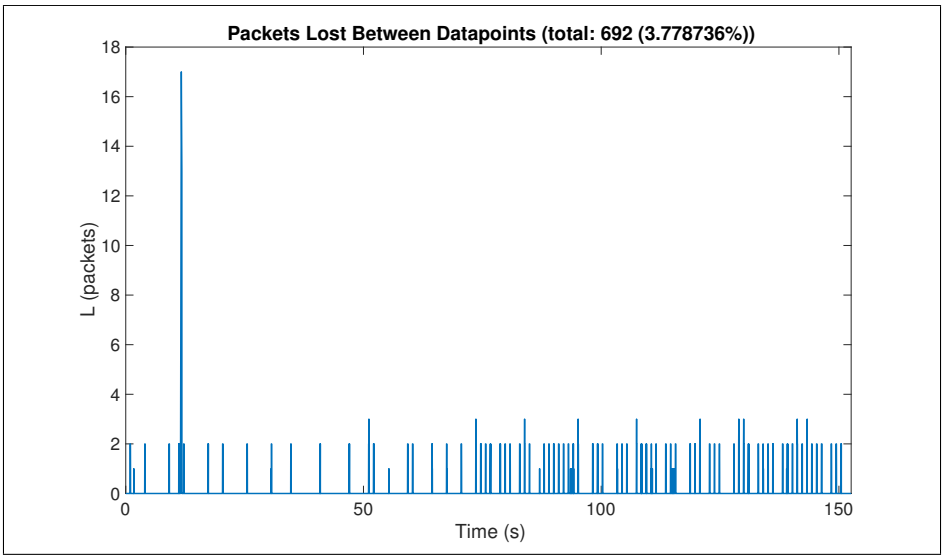


Figure 3: Packet losses over a two-and-a-half-minute test, with receiver on the ACSPG sampling at 1,000 Hz and transmitter sending at 120 Hz.

3.2 Data Packaging and Compression

There are a number of methods to compress quaternion data, one of the simplest of which is known as "smallest three" compression (Algorithm 1)[7]. In this method, only the smallest three components of the quaternion are transmitted, along with a two-bit number to tell the receiver which component was removed, and the knowledge that the quaternion must have a norm of one is used to reconstruct it. Smallest three compression allows for smaller packet sizes than sending all four components directly, and can in fact give higher precision because, as can be demonstrated mathemetically, the smallest three components range between $\pm\frac{\sqrt{2}}{2}$, as opposed to ± 1 if the largest component is included[7].

Table 3: Mean error ($\delta = |q - q_{initial}|$) after compressing and unpacking 100,000 random quaternions using each method.

Method	Smallest Three (Algorithm 1)		No Compression (Algorithm 2)	
Number of Elements	3		4	
Size of Element (bits)	32	64	32	64
Mean Error (unitless)	$4 \cdot 10^{-10}$	$0 \cdot 10^{-10}$	$14 \cdot 10^{-10}$	$0 \cdot 10^{-10}$

As Table 3 shows, a series of tests in which uniform random quaternions were generated, compressed, bitpacked and unpacked indicated that the best performance would be achieved using 64 bit elements. Using 64 bits for each quaternion component would significantly increase packet size, however, and the error level was not seen to be significant with any compression scheme. Of the two 32 bit schemes, smallest three incurred the least error, but it was decided

to proceed with the simpler, four element approach (Algorithm 2), in order to minimize the complexity needed in the Simulink-based receiver.

Algorithm 1 Smallest Three Compression

Require: unit quaternion q
Ensure: buffer of s-bit unsigned integers b and index i of missing component

procedure SMALLEST_THREE_PACKING(q)
  $i \leftarrow 1$
  for $k = 2$ to 4 **do**
    if $|q_k| > |q_i|$ **then**
      $i \leftarrow k$
    end if
  end for
  if $q_i < 0$ **then**
    $q \leftarrow -q$
  end if
  $n \leftarrow 0$
  for $k = 1$ to 4 **do**
    if $k \neq i$ **then**
      $b_n \leftarrow 2^{s-1}(q_k \frac{\sqrt{2}}{2} + 1) - 1$
      $n \leftarrow n + 1$
    end if
  end for
  send b and i
end procedure

Algorithm 2 Basic Quaternion Bitpacking

Require: unit quaternion q
Ensure: buffer of s-bit unsigned integers b

procedure BASIC_PACKING(q)
  for $k = 1$ to 4 **do**
    $b_k \leftarrow 2^{s-1}(q_k + 1) - 1$
  end for
  send b
end procedure

Beyond compressing the critical quaternion data, each packet needs to include at least one more value to facillitate the planned system diagnostic tools. The initial design included a timestamp (t) and a packet index number (i), while the final design added the PhaseSpace system's "confidence" output (c), which represents the quality of the quaternion estimate. The decision process behind the inclusion of these additional data, as well as how they are used, is detailed in Section 3.4.

3.3 Simulink Model Design for SpeedGoat

One of the key features of the ACSPG is that it allows the user to develop software for it directly in Simulink thanks to its SpeedGoat computer. To provide the user with motion capture data in real time, a Simulink subsystem was developed to receive and unpack these data, making them available to log and distribute to the ADCS. The initial design of the diagnostic software was based on a response system, in which the receivers would send reply packets to the transmitter each time a data packet arrived, with the intention that this could be used to evaluate round-trip time (RTT) and to simplify data logging. Subsequent testing revealed that using Simulink's *UDP Send* and *UDP Receive* blocks at the same time caused periodic (about once per second) dropouts of 2–3 tenths of a second, during which no data were received by the ACSPG. As a result, the *UDP Send* block was removed and the system now logs data directly onto the ACSPG's flight computer (see Section 3.4 for more information).

Beyond the periodic droupout issue, the Simulink model design is also made more complicated by the interplay between the real-time components of the model and Simulink's fixed sampling rate. A *While Iterator Subsystem* is used to ensure that all data received between Simulink steps are removed from the UDP buffer and processed. In practice, however, comparisons between sampling rates and the data logged by the model indicate that only the last packet processed in each step is actually distributed through the model. This is not necessarily a problem, however, because when packets are arriving faster than the receiver is reading them, there should still be a new, up-to-date packet for almost every Simulink step.

The data logged by the Simulink model also appear to be interleaved, with the first 50 or so datapoints in the log file actually covering every n^{th} packet from the beginning to almost the end of the time range, followed by the 50 $(n+1)^{\text{th}}$ packets, the 50 $(n+2)^{\text{th}}$ packets and so on. This is most likely due to an incorrect setting in the file scope used to store the data, but it has proven simple enough to reorder the data by timestamp in post-processing.

3.4 Diagnostic Tools Development

3.4.1 Metrics

Due to the complexity of the motion capture system and the importance that it be able to send data quickly, it was required not only that the software and hardware for data transfer be designed, but also that software features be incorporated at the base level of the system that could be used to evaluate performance and to diagnose problems if they arose in the future. To that end, each data packet created and sent by the transmitter program includes 64 bits containing the UNIX timestamp[8] at which it was created (in microseconds) and an additional 64 bits containing its unique index. It was chosen that these values be stored in 64 bits because with 32 bit packet elements, it was the smallest number of elements that would allow the system to produce properly-numbered packets for a reasonable amount of time. At one element (32 bits), the timestamp would overflow in $\frac{(2^{32}-1)\cdot 10^{-6}}{3600} = 1.19$ hours, while at two elements (64 bits), it would last for $\frac{(2^{64}-1)\cdot 10^{-6}}{3600} = 5.12 \cdot 10^{9}$ hours of operation.

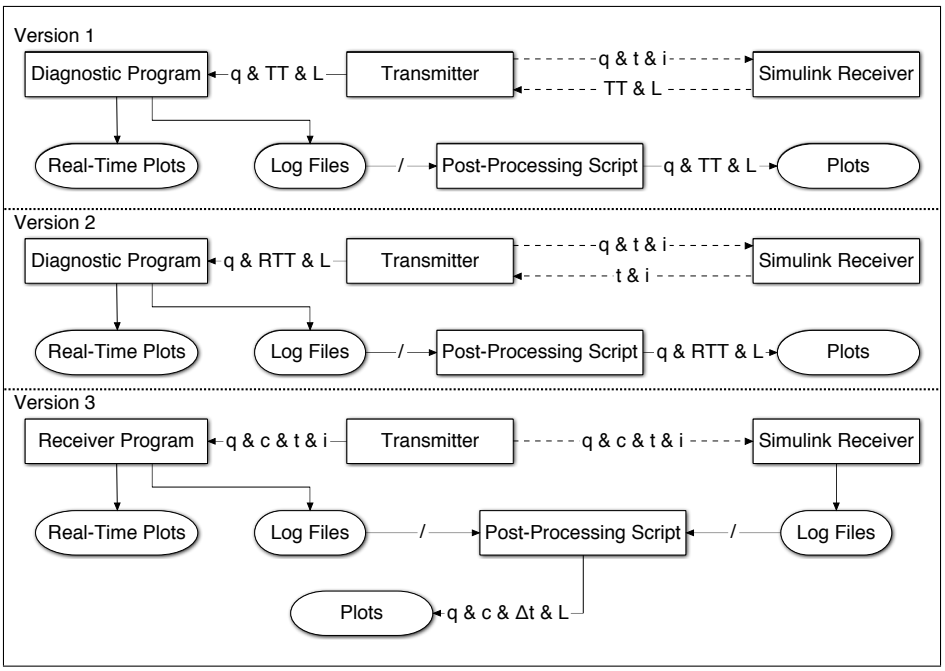


Figure 4: Three generations of the diagnostic tools. Key: dashed lines are wireless communication, solid lines are physical, slashes indicate post-processing. Variables: q is the attitude quaternion, c is the confidence of q, TT is duration of travel (latency), RTT is round-trip time, Δt is the time between packets, L is the number of packets lost since the previous packet, i is the packet's index and t is its timestamp.

Initially, the timestamp and index numbers were intended to be processed by the receiver. The timestamp would

be compared to the current clock time to determine the time duration it had taken to arrive and the index would be compared to that of the previous packet to determine if any packets had been lost. The duration of travel (TT) and loss (L) would then be packaged into a response packet and sent back to the transmitter for logging. This method proved impossible with the computer hardware, because the clock times and speeds varied by too much from device to device to achieve reliable timing, so the design was revised.

Under the new diagnostic setup, the timestamp and index would be sent back directly in the response packet. This would make it impossible to measure the duration of travel directly, but timestamps could be compared to the transmitter's clock to determine round-trip time (RTT). The RTT for a packet could be used to approximate its TT by assuming that both legs of the trip take the same amount of time, and would at least provide an upper bound. Through further examination of the design, however, it was determined that RTT was not as important as the rate at which new packets were received and the consistency of that rate. The rationale behind this was that while the duration of travel could introduce a delay into the system, an inconsistent data rate could prevent new packets from being available when needed and potentially reduce the stability of the attitude estimate. RTT was subsequently replaced as a metric by Δt, the difference in timestamps between consecutive packets received. This proved to be a fortunate design decision because, like packet loss, Δt can be calculated on the receiver without sending responses back to the transmitter, which is not possible due to the *UDP Send* issue discussed in Section 3.3.

The other metric included in the diagnostic portion of the data packet is c, the Impulse X2 system's confidence in the quaternion output, which is the reciprocal condition number of the covariance matrix in the filter that is applied to the raw camera data[1]. While the exact significance of this metric is difficult to quantify, nominal values can be determined through repeated testing (typically varying between 120 and 200 for our system, at its maximum when the platform is horizontal), which can facilitate quicker problem identification and be used to determine whether changes to the system have a positive or negative impact on performance. A confidence plot over the range of motion of the ACSPG can be found in Figure 6 of Section 4.1.

3.4.2 Plotting and Analysis Tools

The initial software design (Figure 4) included a diagnostic receiver program, which would be run on the workstation that is physically connected to the Owl server and would receive network statistics based on the receiver's response packets and quaternion data. These statistics and attitude data would be plotted in real time to help the user monitor the system during tests and would also be logged to files for later plotting and analysis. After the transition from the response-based setup to direct data logging on the SpeedGoat computer, this program (outputs shown in Figure 5) evolved into a real-time visualizer and a quaternion data logger, providing an independent log of the motion capture system's performance without being hampered by wireless communication losses and delays.

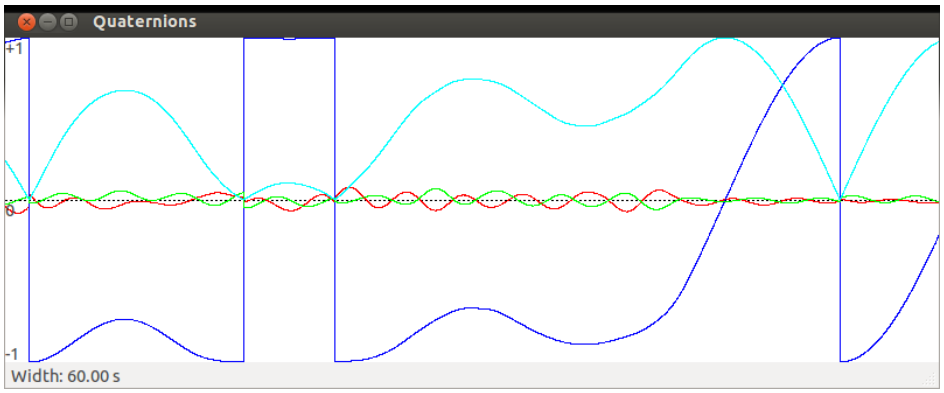
In addition to the diagnostic program, the software suite also includes a set of MATLAB scripts that allow the user to retrieve logged data from the ACSPG or the diagnostic workstation and perform statistical analyses and produce plots. These scripts were used to generate most of the figures in this document.

4 Sample Results

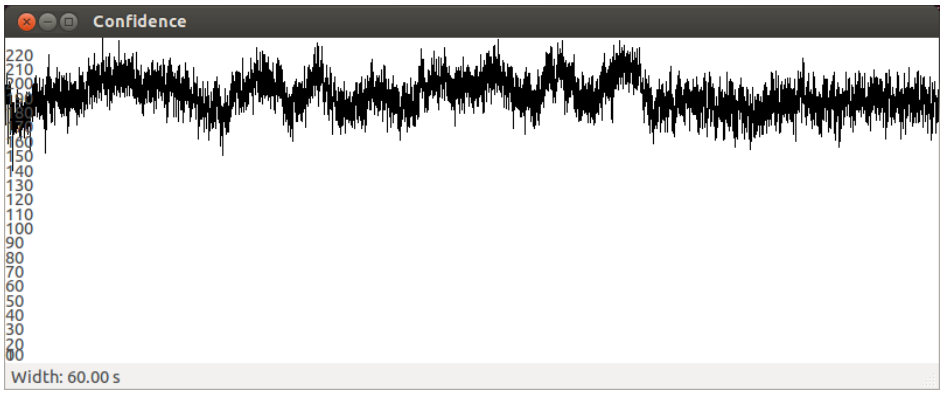
4.1 Range of Motion Coverage

Figure 6 shows motion capture data collected at 480 Hz by the Impulse X2 system. During data collection, the ACSPG platform was rotated through a wide range of attitudes, covering its full 180° yaw range and almost its full $\pm 30°$ pitch/roll range. As Figure 6 shows, the quaternion remains smooth and the confidence value, though exhibiting some oscillation, remains consistently above about 120 regardless of platform orientation.

[1] From email communication with PhaseSpace.



(a) Real-time plot of q (the attitude quaternion).



(b) Real-time plot of c (the confidence of the quaternion). Higher values are better.

Figure 5: Simultaneous screenshots of real-time data plotting interface, showing 60 seconds of data. Flipping points in the quaternion trajectory are caused by the equivalence between positive and negative quaternions.

4.2 Data Transfer Performance

Figure 7 shows the Δt betwen packets received by the ASCPG, followed by its 10-second window running mean and 1-σ bounds, then the number of packets lost between each datapoint. These data were collected with the receiver sampling for packets at 1,000 Hz and the transmitter sending them at its maximum rate (limited by the motion capture system) of 480 Hz. These plots demonstrate that the system is capable of a high and consistent rate of data transfer, with the mean rate always near 480 Hz and the standard deviation of Δt always less than 2 ms. They also show that the rate of packet loss is only 6.47%, further validating the decision to use the lossy UDP protocol in the design of the system.

5 Future Work

One important step in continuing to develop this system is to perform tests to quantify the accuracy and noise in the quaternion measurements produced by the motion capture system, more thoroughly demonstrating its effectiveness. Relatedly, the confidence metric mentioned in Section 3.4.1 and used in Section 4.1 to verify attitude-independent performance may have some utility in attitude filtering if the relationship between a given sample's confidence and the reliability of its quaternion can be characterized. Finally, the packet creation and unpacking code could be modified to use more efficient bitpacking methods, for example by using the smallest three compression mentioned in Section 3.2 and by using the minimum amount of bits needed for each packet element rather than working in 32-bit increments.

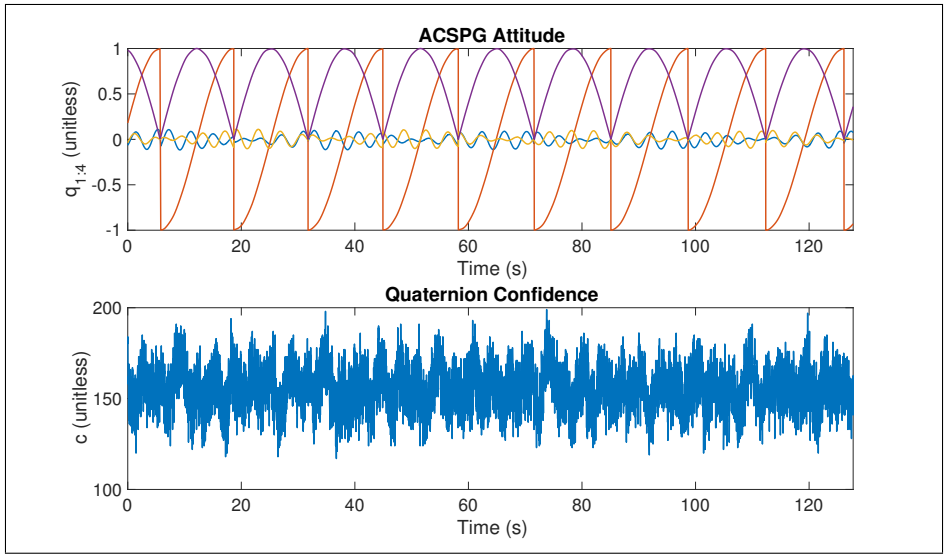


Figure 6: Two minutes of motion capture data, showing quaternion components and confidence. Captured at 480 Hz.

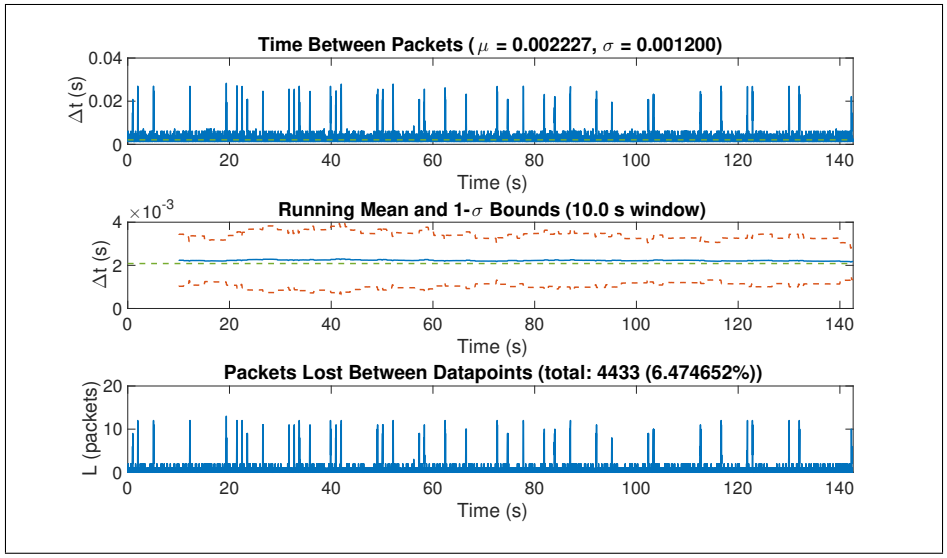


Figure 7: Value of Δt for each packet received, its running mean and 1-σ bounds (10 s window), and packet loss. Data transmitted at 480 Hz, UDP receiver sampling at 1,000 Hz. The dashed green line in the first two plots represents one 480^{th} of a second, the expected value of Δt.

6 Conclusion

The ACSPG's ability to receive inertial attitude information in real time is critical to both effective attitude sensor simulation and the evaluation of attitude control performance. The work described here has made it possible to reliably transfer such data from a motion capture system to the ACSPG's onboard computer. The system has demonstrated a promising degree of fidelity and a high rate of data transmission with a sufficiently low rate of data loss. The software architecture that has been designed will also provide future users of the ACSPG testbed tools to further evaluate performance and diagnose any problems that may arise.

7 Acknowledgments

I would like to thank my mentor, 1st Lt Dylan Penn, for his help and support, as well as Steve Ortega, David Hill and Kenneth Elliot of Applied Technology Associates from the ACSPG team for their techincal expertise on this project. I would also like to thank the AFRL Scholars program for the opportunity to participate in this research.

References

[1] Wiley J. Larson and James R. Wertz. *Space Mission Analysis and Design*, volume 8 of *Space Technology Library*. Microcosm Press, Torrence, CA, 3 edition, 1999. ISBN 9781881883104.

[2] Jana L. Schwartz, Mason A. Peck, and Christopher D. Hall. Historical Review of Air-Bearing Spacecraft Simulators. *Journal of Guidance, Control, and Dynamics*, 26(4):513–522, July–August 2003. doi: 10.2514/2.5085.

[3] Jorge G. Padro. Development of a Star Tracker-Based Reference System for Accurate Attitude Determination of a Simulated Spacecraft. *Defense Technical Information Center*, March 2012. Accession Number: ADA559975.

[4] PhaseSpace, Inc. PhaseSpace Motion Capture. `http://www.phasespace.com`, 2013. Accessed: 2016-08-05.

[5] SpeedGoat GmbH. SpeedGoat - x86 / FPGA real-time hardware for Simulink. `http://www.speedgoat.ch`, 2016. Accessed: 2016-08-05.

[6] Santosh Kumar and Sonam Rai. Survey on Transport Layer Protocols: TCP & UDP. *International Journal of Computer Applications*, 46(7):20–25, May 2012. doi: 10.5120/6920-9285.

[7] Glenn Fidler. Snapshot Compression. `http://gafferongames.com/networked-physics/snapshot-compression/`, 2015. Accessed: 2016-08-10.

[8] The IEEE and The Open Group. IEEE Standard 1003.1, 2013.